02

Recibiendo datos

Transcripción

[00:00] Bien. Ya tenemos configurado Insomnia, ya nuestro cliente está listo para mandar request contra nuestro API. Vamos a ver por el lado del API qué es lo que tenemos que hacer aquí. Si recuerdan la clase pasada, lo que hicimos primero, ¿qué fue? Tenemos que crear el controller para recibir el nuevo request que está en el punto de aquí, en el path de aquí médicos.

[00:22] ¿Para eso qué tenemos que hacer? Crear un MedicoController. Entonces vamos aquí a new, java class y vamos a decirle MedicoController. ¿Por qué MedicoController? Porque vamos a guardar médicos en este caso. Es una clase. La creamos. Aún no la vamos a agregar a Git.

[00:40] Y aquí, ¿qué es lo segundo que tenemos que hacer para decirle a spring, que esos es un controller? Correcto. RestController, lo dejaremos aquí. Y si queremos mapear el path médicos en este controller, ¿qué es lo otro que tenemos que hacer? Agregar el RequestMapping, esto de aquí.

[01:01] Y entre paréntesis el path que queremos mapear, en este caso, "/médicos". Y bien, en un inicio, esto sería todo el setup que deberíamos hacer para mapear un deadpoint. Pero eso no es todo. ¿Por qué? Porque necesitamos un método dónde hacer el handling, atrapar este request que está llegando.

[01:25] Tenemos el Controller, tenemos médicos, pero hasta ahora no hay nada que pueda procesar este endpoint. Para eso creemos un public. Por ahora vamos a dejarlo como un void, registrarMedico. Sin parámetros por el

momento. Y como ya ustedes se imaginan, si yo quiero mapear un request del tipo post como el que tengo aquí, ¿qué tengo que hacer aquí? Exacto.

[01:53] La notación PostMapping. Y automáticamente yo ya tengo el PostMapping de registrarMedicos aquí. No está haciendo nada por el momento pero como yo quiero validar que mi request está siendo exitoso, lo que yo voy a hacer aquí es usar un System.out.println y voy a imprimir aquí: “El request llega correctamente”.

[02:19] Okay, vamos a guardar, guardamos aquí. Esperamos que recargue nuestro servidor porque estamos con Spring DevTools, vemos que ya recargó y vamos a hacer la prueba, enviamos y recibimos un 200. Okay, 200 significa que nuestro request fue exitoso Entonces sí llegó al endpoint, hizo el mapeo correctamente y vemos aquí que el request llega correctamente.

[02:46] Entonces ya tenemos la comunicación entre nuestro cliente, que en este caso es Insomnia, recuerden, puede ser una aplicación front end, incluso mobile, puede ser Postman, lo que ustedes quieran pero ya estamos consiguiendo esta comunicación entre el cliente y el API, nuestro back end, ya estamos por buen camino. Pero sin embargo hay algo que estamos olvidando aquí y son los datos, lo que de verdad nos importa.

[03:15] Nosotros queremos guardar estos datos en nuestro API. ¿Cómo podemos recibir estos datos? Pues bien. Aquí en registrarMedico lo que tenemos que hacer es agregar un parámetro a este método que le diga los datos que están siendo enviados desde el cliente. Entonces por ahora yo lo voy a declarar todo como un string porque lo que tenemos aquí es un JSON, que puede ser tranquilamente un string JSON.

[03:43] Entonces le voy a poner aquí string parámetro. Entonces este parámetro es lo que debería llegar de este JSON hacia mí método registrarMedico, entonces segundo yo voy a imprimir ese parámetro para validar que los datos estén llegando correctamente. Nuevamente guardo.

Espero a que Spring DevTools haga su trabajo de reiniciar el servidor por mí. Y vemos que ya fue hecho.

[04:17] Volvemos a Insomnia y damos a enviar. Nuevamente vemos que nos retorna un 200. Vemos que el request llega correctamente, pero aquí tenemos null en parámetro. Entonces por alguna razón, estos datos no están siendo enviados como este parámetro.

[04:42] Aquí hay un problema y es que así como tú le dices a Spring, Spring por si acaso este es un controller, al momento que cargas el contexto, considéralo como un controller. El Spring por si acaso, el end point de este Controller va a ser médicos. Spring por si acaso este método va agarrar un post. No le hemos dicho a Spring que este parámetro es el body del request.

[05:08] Y para eso, típico de Spring, tenemos que usar una notación llamada requestBody. Es la de aquí, se la agregamos y ahora Spring debería ser capaz de reconocer este parámetro, porque ya le estamos diciendo por si acaso eso es el body que está llegando de nuestro request.

[05:30] Entonces vamos a limpiar aquí en la terminal, venimos aquí y vamos a enviar nuevamente el request. Vemos que da un 200, y en efecto, si ustedes pueden ver aquí, ya tenemos que request llega correctamente y miren lo que está aquí abajo. Tenemos el mismo JSON que nosotros enviamos aquí a nivel del request.

[05:54] Entonces hasta ahora ya tenemos dos puntos cubiertos. El primero, crear nuestro cliente; el segundo, obtener los datos que nuestro cliente ha enviado y ya tenerlos a nivel del API de nuestro back end. Ahora, por ejemplo, ¿qué sucedería si yo quiero trabajar solamente a nivel del nombre del Médic? En este caso, yo quiero validar que el médico no existe por lo tanto necesito el nombre para buscar en alguna base datos y ver si el médico existe o no.

[06:25] En ese caso, si trabajas con un string tendrías que hacer un tipo de substring para extraer este valor de campo nombre y extraer aquí el valor que

tiene ese campo que sería Rodrigo López en este caso.

[06:48] Eso no es lo más recomendable porque trabajar con string, aparte que performáticamente no es óptimo porque los strings no se modifican, recuerdan son siempre mutables, entonces la máquina virtual de Java tendría que constantemente crear nuevos objetos en la memoria.

[07:04] También es tedioso nivel de programación. En el siguiente video, vamos a ver ya una forma más optimizada, más recomendada, de tratar con objetos que están llegando a nuestro API de modo que yo podría decir algo así como "imprímeme el nombre que está llegando en este parámetro", por ejemplo, tratarlo ya a nivel de objetos. Nos vemos en el siguiente video.